Module:3 Perform in Structured Query Language (SQL) program.

Unit-1: Familiarize with Structured Query Language (SQL)

What is SQL? *(SQL भनेको के हो ?)*

SQL को पूरा रुप Structured Query Language हो । यो एउटा प्रोग्रामिङ ल्याङग्वेज हो, जसले रिलेसनल डाटावेसमा रहेका डाटालाई क्रमबद्ध गर्ने, परिवर्तन गर्ने, पुनः प्राप्त गर्ने कार्य गर्दछ ।

SQL एउटा रिलेसनल डाटावेस सिस्टमको लागि स्तरीय भाषा हो । धेरैजसो रिलेसनल डाटावेस सिस्टमहरु जस्तै : MySQL, MS Access, Oracle, Sybase, Informix, postgres, SQL Server ले SQL लाई स्तरीय भाषाको रुपमा प्रयोग गरेको पाइन्छ ।

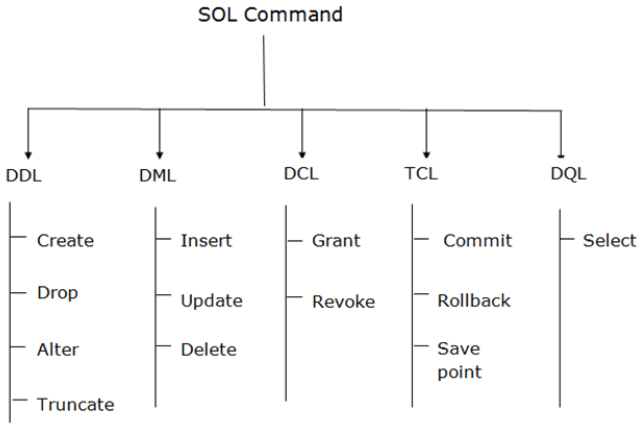
यी प्रोग्रामिङ ल्याङग्वेजहरुले विभिन्न उपभाषा प्रयोग गरेको पाइन्छ । जस्तै :

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

Why SQL? *(SQL नै किन ?)*

- यसले प्रयोगकर्तालाई रिलेसनल डाटावेस सिस्टममा डाटाहरु प्रयोग गर्न मद्दत गर्दछ ।
- यसले डाटालाई वर्णन गर्न मद्दत गर्दछ ।
- यसले डाटालाई परिभाषित गर्न तथा उक्त डाटामा हेरफेर गर्न मद्दत गर्दछ ।
- यसले अन्य भाषाहरु जस्तै : SQL modules, libraries & pre-compilers संग मिलाएर राख्न मद्दत गर्दछ । साथै डाटावेस बनाउन, राख्न तथा टेवलहरु बनाउन मद्दत गर्दछ ।
- यसले प्रयोगकर्तालाई डाटावेसको प्रोसिडर तथा फङ्सनहरु हेर्न मद्दत गर्दछ ।
- यसले प्रयोगकर्तालाई कुनैपनि टेवल, प्रोसिडर तथा भ्यूमा अनुमति सेट गर्न मद्दत गर्दछ ।

Unit-2: Design using the Structure SQL



Data-Manipulation Language

Data-manipulation language (DML) एउटा ल्याङ्ग्वेज (भाषा) हो जसले प्रयोगकर्तालाई उचित data model अनुसार डाटालाई प्रयोग गर्न तथा हेरफेर गर्न मद्दत गर्दछ । साधारणतया यी दुई किसिमका छन् :

- Procedural DML: यसमा प्रयोगकर्ताले कस्तो डाटा कसरी प्राप्त गर्ने भन्ने कुरा किटान गर्नुपर्ने हुन्छ ।
- Declarative DMLs (also referred to as nonprocedural DMLs): यसमा प्रयोगकर्ताले कस्तो डाटा कसरी प्राप्त गर्ने भन्ने कुरा किटान गर्नुपर्ने हुदैन ।

Data manipulation is used for the following purposes: (यसका उद्देश्यहरु)

- डाटावेसमा रहेका सूचनाहरु पुनः प्राप्त गर्नु
- डाटावेसमा नयाँ डाटा थप्नु
- अनावश्यक डाटा डाटावेसबाट हटाउनु
- आवश्यक डाटा सच्चाउनु

Data- Definition Language

data-definition language (DDL) एउटा विशेष किसिमको भाषा हो जसको सहायताले हामीले डाटावेस schema लाई परिभाषित गर्न सक्छौं ।

जस्तै : दिइएको SQL स्टेटमेन्टले एउटा एकाउण्ट टेवल बनाउँदछ ।

```
create table account
(
account-number char(10),
```

balance integer
);

माथि दिइएको (DDL) स्टेटमेन्टले एउटा एकाउण्ट टेवल बनाउँछ । साथसाथै यसले विशेष टेवलहरुको सेटलाई पनि अपडेट गर्दछ जसलाई data dictionary or data directory भनिन्छ ।

Functions of DDL (यसका कार्यहरु)

हामिले विशेष प्रकारको डाटावेसको DDL स्टेटमेन्टहरु प्रयोग गरेर डाटावेस भण्डार संरचना तथा एस्सेस गर्ने तरिकाहरु निर्धारण गर्न सक्छौँ । यिनी स्टेटमेन्टहरुले डाटावेस स्किमाहरुको विस्तृत कार्यान्वयन गर्न सहयोग गर्दछ जुन सामान्यतया प्रयोगकर्ताबाट लुकेको हुन्छ ।

SQL Commands:

रिलेसनल डाटाबेसमा काम गर्दा सामायन्यता प्रयोग हुने कमाण्डहरु : CREATE, SELECT, INSERT, UPDATE,DELETE and DROP हुन् । यी कमाण्डहरुलाई यिनीहरुको प्रकृति अनुसार विभिन्न समूहमा वर्गिकरण गरिएको छ ।

DDL - Data Definition Language:

Command Description

CREATE: यस कमाण्डले डाटावेसमा नयाँ टेवल बनाउने, टेवलको संरचना देखाउने वा अन्य object बनाउन मद्दत गर्दछ ।

ALTER: यस कमाण्डले डाटावेसमा रहेका object हरु परिवर्तन गर्दछ जस्तै : टेवल ।

DROP: यस कमाण्डले डाटावेसमा रहेका टेवल, टेवलको संरचना वा अन्य object हटाउन मद्दत गर्दछ ।

DML - Data Manipulation Language:

Command Description

INSERT: यस कमाण्डले बनेको टेबलमा नयाँ रेकर्डलाई राख्न मद्दत गर्दछ ।

UPDATE: टेबलमा रहेको रेकर्डहरु परिवर्तन गर्ने कार्यमा यस कमाण्डको प्रयोग हुन्छ ।

DELETE: टेबलमा रहेको रेकर्डहरु हटाउने कार्यमा यस कमाण्डको प्रयोग हुन्छ ।

DCL - Data Control Language:

Command: Description

GRANT: युजरलाई विशेष अधिकार प्रदान गर्ने कार्यमा यस कमाण्डको प्रयोग गरिन्छ ।

REVOKE: युजरको विशेष अधिकार हटाउने कार्यमा यस कमाण्डको प्रयोग गरिन्छ ।

DQL - Data Query Language:

Command Description

SELECT: एक वा एकभन्दा वढी टेवलहरुबाट निश्चित रेकर्डहरु प्राप्त गर्ने कार्यमा यस कमाण्डको प्रयोग गरिन्छ ।

Reserved words and reserved characters:

Within SQL certain words are *reserved*. You cannot use an SQL reserved word as an SQL identifier (such as the name for a table, a column, an AS alias, or other entity), unless:

ADD, ALL, ALLOCATE, ALTER, AND, ANY, ARE, AS, ASC, ASSERTION, ATAUTHORIZATION etc.

Capital and small character's in SQL:

SQL UPPER Function

We use SQL UPPER function to convert the characters in the expression into uppercase. It converts all characters into capital letters.

The syntax of SQL Upper function is:

```
SELECT UPPER (expression) FROM [Source Data]
```

Let's use some examples for this function and view the output.

Example 1: Use UPPER function with all lower-case characters in a single word

```
SELECT UPPER('sqlshack');
```

It gives the following output.

```
SELECT UPPER('sqlshack');
```

SQLSHACK

Example 2: Use UPPER function with all lower-case characters in an expression

In this example, we use a string with Upper function, and it converts all letters in the string to uppercase.

```
SELECT UPPER ('learn sql server with sqlshack');
```

Output: It converts all characters for a string.

```
SELECT UPPER('learn sql server with sqlshack');
```

LEARN SQL SERVER WITH SQLSHACK

Example 3: Use an SQL UPPER function with mix case (combination of the lower and upper case) characters in an expression

In this example, we have the same string from example 2 but with a combination of lower- and upper-case characters.

```
SELECT UPPER('Learn SQL server with sqlshack');
```

It converts all characters regardless of lower- and upper-case characters.

```
SELECT UPPER('Learn SQL server with sqlshack');
```

LEARN SQL SERVER WITH SQLSHACK

Example 4: Use the UPPER function with all uppercase characters

In the case of an upper case letter, this function does not perform any operation and return the same string as output.

```
SELECT UPPER('LEARN SQL SERVER WITH SQLSHACK');
```

Output:

```
SELECT UPPER('LEARN SQL SERVER WITH SQLSHACK');
```

LEARN SQL SERVER WITH SQLSHACK

Example 5: Use an SQL UPPER function with Select statement

We can use SQL UPPER function in a select statement as well and convert the required values to upper case.
In the following query, it creates an employee table and inserts record in it.

```
Create table Employee
(
  Firstname varchar(20),
  Lastname varchar(20),
  Country varchar(20)
)

Insert into Employee values ('Rajendra','Gupta','India')
```

Perform a select query on this employee table, and it returns the records in the following format.

```
SELECT Firstname,
      Lastname,
```

```
    Country
FROM Employee;
```

	Firstname	Lastname	Country
1	Rajendra	Gupta	India

We want values in the country column to be in uppercase. Let's use the Upper function.

```
SELECT Firstname,
    Lastname,
    upper(Country) as COUNTRY
FROM Employee;
```

In the output, we can see the uppercase value for a Country column.

	Firstname	Lastname	COUNTRY
1	Rajendra	Gupta	INDIA

upper(Country) as COUNTRY

Example 6: Use an UPPER function with an update statement

We can use this function in an update statement as well. The following query, update an employee table record with uppercase of the [Firstname] column value.

```
Update [dbo].[Employee] set [Firstname] =upper('raj') where ID=1
```

Example 7: Use SQL UPPER function a variable

We can use a variable in T-SQL as well. We can use the upper function with a variable as well.
In the following query, we declare a variable and provide string in it.

```
DECLARE @text VARCHAR(30);
SET @text = 'This is a sample text';
SELECT @text as Input,
    UPPER(@text) AS UpperCase;
```

	Input	UpperCase
1	This is a sample text	THIS IS A SAMPLE TEXT

SQL LOWER function

It works opposite to the SQL UPPER function. It converts uppercase letters to lower case for the specified text or expression. The syntax of SQL Lower function is

```
SELECT Lower (Expression) FROM Source
```

Let's use some examples and view the output of this function.

Example 1: Use a LOWER function with all lower-case characters in a single word

In this example, we use a lower function with all lower-case characters. It does not perform any character case conversion for this.

```
SELECT Lower('sqlshack');
```

```
SELECT Lower('sqlshack');
```

sqlshack

Example 2: Use SQL Lower function with all lower-case characters in an expression

In this example, we have a string with all lower-case characters. We get the same output because of all character already in lower case.

```
SELECT Lower('learn sql server with sqlshack');
```

learn sql server with sqlshack

Example 3: Use a LOWER function with mix case (combination of the lower and upper case) characters in an expression

In this example, we have a string that contains both lower and upper case. SQL Lower function ignores the lower characters and converts all uppercase characters into lowercase.

```
SELECT Lower('Learn SQL server with sqlshack');
```

```
SELECT Lower('Learn SQL server with sqlshack');
```

learn sql server with sqlshack

Example 4: Use a Lower function with all uppercase characters

Suppose we have a string with all uppercase letters, and we require converting them into lowercase. SQL Lower function does it for us.

```
SELECT Lower('LEARN SQL SERVER WITH SQLSHACK');
```

```
SELECT Lower('LEARN SQL SERVER WITH SQLSHACK');
```

learn sql server with sqlshack

Example 5: Use a LOWER function with Select statement

In the following example, we use SQL Lower function to convert the [firstname] column values in lowercase.

```
SELECT Firstname,
    Lastname,
    lower(Country) as COUNTRY
FROM Employee;
```

	Firstname	Lastname	COUNTRY
1	Rajendra	Gupta	india

Example 6: Use SQL LOWER function with an update statement

We can use this function in an update statement as well. The following query, update an employee table record with lowercase of the Lastname column.

```
Update [dbo].[Employee] set [lastname]=Lower('RAJ')
```

Example 7: Use the LOWER function in a variable

We can use a lower function with a variable similar to an upper function. Let's use the same query with lower function.

```
DECLARE @text VARCHAR(30);
SET @text = 'This is a sample text';
SELECT @text,
    LOWER(@text) AS Lowercase;
```

SQL RDBMS Concepts

What is RDBMS?

RDBMS को पूरा रुप Relational Database Management System हो । RDBMS नै SQL र अन्य डाटावेस सिस्टमहरु जस्तै : MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access को आधार हो ।

RDBMS यस्तो database management system DBMS हो जुन E. F. Codd ले प्रतिपादन गरेको रिलेसनल मोडेलमा आधारित छ ।

What is table?

रिलेसनल डाटावेसमा डाटाहरु भण्डार गर्न प्रयोग गरिने डाटावेस object लाई टेवल भनिन्छ । टेवल भनेको सम्बन्धित डाटाहरुको समूह हो जसमा डाटाहरु रो र कोलमको रुपमा मिलाएर राखिएको हुन्छ । रिलेसनल डाटावेसमा टेवल सबैभन्दा सजिलो तथा सामान्य रुपमा डाटा भण्डार गर्न प्रयोग गरिन्छ ।

तल दिइएको उदाहरण एउटा Employee टेवलको उदाहरण हो ।

```
+----+----------+-----+---------------+--------------+
| ID | NAME     | AGE | ADDRESS       | SALARY       |
+----+----------+-----+-----------+----------+
| 1  | Ramesh   | 32  | Kathmandu | 2000.00 |
| 2  | Khilan   | 25  | Dang      | 1500.00 |
| 3  | Kaushik  | 23  | Rolpa     | 2000.00|
| 4  | Chaitali | 25  | Pyuthan   | 6500.00 |
| 5  | Hardik   | 27  | Rukum     | 8500.00 |
+----+----------+-----+------------------+-----------  +
```

What is field?

प्रत्येक टेवललाई विभिन्न स-साना entity मा विभाजन गरिएको हुन्छ जसलाई field भनिन्छ । माथि दिइएको Employee टेवलमा रहेका ID,NAME, AGE, ADDRESS and SALARY फिल्ड हुन् ।
फिल्ड भनेको टेवलमा रहेको कोलम हो जसमा प्रत्येक रेकर्डहरुको एउटा निश्चित किसिमको सूचना रहेको हुन्छ ।

What is record or row?

रेकर्ड भनेको टेवलको रो हो जसमा व्यक्तिगत विवरण रहेको हुन्छ । जस्तै माथि दिइएको टेवलमा ५ वटा रेकर्डहरु रहेका छन् । तल दिइएको Employee टेवलको एउटा रेकर्ड हो ।

```
+----+----------+-----+-----------+----------+
| 1  | Ramesh   | 32  | Kathmandu | 2000.00 |
+----+----------+-----+-----------+----------+
```

रेकर्ड भनेको टेवलमा तेर्सो रुपमा रहेको entity हो ।

What is column?

कोलम भनेको टेवलमा ठाडो रुपमा रहेको entity हो जसमा कुनै निश्चित किसिमको सूचना रहेको हुन्छ । जस्तै तल दिइएको Employee टेवलको ADDRESS कोलम हो जसमा विभिन्न ठाउँहरुको सूचना रहेको छ ।

```
+-----------+
| ADDRESS   |
+-------------- +
| Ahmedabad |
| Delhi     |
| Kota      |
| Mumbai    |
```

| Bhopal |
+----+------+

Unit-3: Manage Tables

What is NULL value?

NULL value भनेको टवेलमा रहेको त्यस्तो field हो जुन खाली रहेको हुन्छ अर्थात् त्यसमा कुनै डाटा वा value रहेको हुँदैन । यो कुरा बुझ्न जरुरी छ कि NULL value भनेको zero value वा फिल्डमा रहेको spaces भन्दा फरक हो । NULL value रहेको फिल्ड भनेको त्यस्तो फिल्ड हो जुन रेकर्ड राख्ने बेलामा खाली राखिएको हुन्छ ।

SQL Constraints:

Constraints भनेको टेवलमा रहेका डाटा कोलमहरुको नियम हुन् । यसको सहयोगले कुन कोलममा कस्तो किसिमको डाटा मात्र राख्न सकिन्छ भन्ने यकिन गर्दछ । यसबाट डाटावेसमा रहेको टेवलको डाटाहरुमा शुद्धता र विश्वसनीयता कायम गर्न सकिन्छ । Constraints पनि दुई किसिमका हुन्छन् be column level or table level । Column level constraints हरुले कुनै एउटा कोलमको नियमहरु निर्धारण गर्दछ भने table level constraints हरुले एउटा टेवलको नियमहरु निर्धारण गर्दछ । तल दिइएका SQL मा सामान्यतया उपलब्ध constraints हुन् ।

- NOT NULL Constraint: यसले कुनैपनि कोलममा NULL value छैन भन्ने सुनिश्चित गर्दछ ।
- DEFAULT Constraint: यसले कोलममा कुनै value राखिएन भने default value प्रदान गर्दछ ।
- UNIQUE Constraint: यसले कोलममा रहेका प्रत्येक value फरक छन् भन्ने सुनिश्चित गर्दछ ।
- PRIMARY Key: यसले डाटावेस टेवलमा रहेका प्रत्येक रेकर्ड वा रो लाई पत्ता लगाउँदछ ।
- FOREIGN Key: यसले अर्को डाटावेस टेवलमा रहेका प्रत्येक रेकर्ड वा रो लाई पत्ता लगाउँदछ ।
- CHECK Constraint: CHECK constraint ले कुनै कोलममा रहेका पत्येक value ले निश्चित conditions प्रयोग गरेको नगरेको यकिन गर्दछ ।
- INDEX: डाटावेसमा डाटा छिटो राख्न वा प्राप्त गर्न मद्दत गर्दछ ।

SQL Data type:

SQL data type भनेको attribute हो जसले कुनैपनि object मा रहेको डाटालाई जनाउँदछ । SQL मा प्रत्येक कोलम, भेरिएवल तथा एक्सप्रेसनको सम्बन्धित data type रहेको हुन्छ । हामीले टेवलहरु बनाउँदा यस्ता डाटा टाइपहरु प्रयोग गर्ने गर्दछौँ । हामीले आवश्यकता अनुसार निश्चित टेवलमा रहेको कोलमको data type छान्ने गर्दछौँ ।

Numeric data type:

DATA TYPE	FROM	TO
Bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
Int	-2,147,483,648	2,147,483,647
Smallint	-32,768	32,767
Tinyint	0	255
Bit	0	1
Decimal	-10^38 +1	10^38 -1
Numeric	-10^38 +1	10^38 -1
Money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
Smallmoney	-214,748.3648	+214,748.3647

Approximate numeric data type:

DATA TYPE	FROM	TO
Float	-1.79E + 308	1.79E + 308
Real	-3.40E + 38	3.40E + 38

Character string data type:

DATA TYPE	FROM	TO
Char	Char	Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
Varchar	Varchar	Maximum of 8,000 characters.(Variable-length non-Unicode data).
varchar(max)	varchar(max)	Maximum length of 231characters, Variable-length non-Unicode data (SQL Server 2005 only).
Text	text	Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

Create table:

सामान्य टेवल बनाउँदा टेवलको नाम, त्यसमा प्रयोग हुने कोलमको नाम तथा कोलमको डाटा टाइप दिनुपर्ने हुन्छ। SQL मा CREATE TABLE statement नयाँ टेवल बनाउन प्रयोग गरिन्छ।

Syntax: CREATE TABLE statement को syntax तल दिइएको छ।

CREATE TABLE table_name(column1 datatype,column2 datatype,column3 datatype, ... columnN datatype,PRIMARY KEY(one or more columns));

SQL CREATE TABLE Example

एउटा टेवल जसको नाम "Persons" र जसमा ५ वटा कोलमहरु : PersonID, LastName, FirstName, Address, and City बनाउनु परेमा निम्न कमाण्ड लेख्नुपर्ने हुन्छ। यस कार्यका लागि CREATE TABLE statement प्रयोग गरिन्छ।

```
CREATE TABLE Persons(
PersonID int,
LastName varchar(255),
FirstNamevarchar(255),
Address varchar(255),
City varchar(255)
 );
```

SQL DROP Table

यदि टेबलको टेबल definition र सबै डाटा, indexes, triggers, constraints, and permission हटाउनु परेमा SQL DROP TABLE statement प्रयोग गरिन्छ।

Syntax:

DROP TABLE statement को Basic syntax निम्न छ।

DROP TABLE table_name;

For example:

यदि CUSTOMERS table हटाउनु परेमा निम्न DROP TABLE statement प्रयोग गर्नुपर्ने हुन्छ।

SQL> DROP TABLE CUSTOMERS;

SQL ALTER Table

The ALTER TABLE statement को प्रयोगले टेवलमा नयाँ कोलम थप्न, हटाउन तथा फेरवदल गर्न सहयोग गर्दछ ।

टेवलमा नयाँ कोलम थप्न निम्न कमाण्ड प्रयोग गर्नुपर्दछ ।

ALTER TABLE table_nameADD column_name datatype;

टेवलको कोलम हटाउनु परेमा निम्न कमाण्ड प्रयोग गर्नुपर्दछ । (कुनै डाटावेस सिस्टमले कोलम हटाउन नदिन सक्छ)

ALTER TABLE table_nameDROP COLUMN column_name;

टेवलको डाटा टाइप परिवर्तन गर्नुपरेमा निम्न कमाण्ड प्रयोग गर्नुपर्दछ ।

ALTER TABLE table_nameALTER COLUMN column_name datatype;

PRIMARY Key:

Primary key यस्तो फिल्ड हो जसले डाटावेस टेवलको प्रत्येक रेकर्डलाई विशेष प्रकारले पत्ता लगाउँदछ । प्राइमरी कि मा जहिले पनि unique values हुनुपर्दछ । प्राइमरी कि मा NULL values हुनुहुदैन । यसमा values हरु दोहोरिएको पनि हुनुहुँदैन ।

यउटा टेवलमा एउटा प्राइमरी कि रहेको हुनुपर्दछ जुन एक वा एक भन्दा बढी फिल्डहरु मिलेर बनेको हुनसक्छ । जब एकभन्दा बढी फिल्डहरु मिलाएर प्राइमरी कि बनाइन्छ भने त्यसलाई composite key भन्ने गरिन्छ । यदि एउटा टेवलको कुनै फिल्डमा प्राइमरी कि सेट गरिएको छ भने त्यस फिल्डमा एउटै value दुईवटा रेकर्डमा हुनुहुँदैन ।

Create Primary Key:

यहाँ CUSTOMERS table मा ID attribute लाई primary key को रुपमा सेट गर्ने syntax दिइएको छ ।

```
CREATE TABLE CUSTOMERS(
ID INT NOT NULL,
NAME VARCHAR (20) NOT NULL,
AGE  INT NOT NULL,
ADDRESS  CHAR (25) ,
SALARY   DECIMAL (18, 2),
PRIMARY KEY (ID)
);
```

OR

यदि CUSTOMERS table पहिले नै बनेको छ र त्यसमा "ID" column मा PRIMARY KEY constraint सेट गर्नु छ भने निम्न SQL syntax प्रयोग गर्न सकिन्छ ।

ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);

FOREIGN Key:

Foreign key त्यस्तो कि हो जसको सहायताले दुईवटा टेवलहरुबिच सम्बन्ध स्थापना गर्न सकिन्छ । यसलाई कहिलेकाहिँ referencing key पनि भन्ने गरिन्छ । Foreign Key एउटा वा एकभन्दा बढी कोलमको समूह हो जसको values अर्को टेवलको Primary Key संग मेल खाएको हुन्छ । कुनै पनि दुईवटा टेवलको बिचमा सम्बन्ध स्थापना गर्दा पहिलो टेवलको प्राइमरी कि संग दोस्रो टेवलको Foreign Key संग मेल खाएको हुनुपर्दछ ।

Example: तल दिइएको दुईवटा टेवलको संरचना बारेमा बिचार गरौँ ।

CUSTOMERS table:

```
CREATE TABLE CUSTOMERS(
ID INT NOT NULL,
```

NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25) ,
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID)
);

ORDERS table:

CREATE TABLE ORDERS (
ID INT NOT NULL,
DATE DATETIME,
CUSTOMER_ID INT references CUSTOMERS(ID),
AMOUNT double,
PRIMARY KEY (ID)
);

OR

यदि ORDERS table पहिले नै बनेको छ र foreign key सेट गरिएको छैन भने निम्न syntax प्रयोग गरेर foreign key सेट गर्नको लागि टेवललाई alter गर्न सकिन्छ ।

ALTER TABLE ORDERS ADD FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (ID);

Unt-4: Edit Data Records

Operate data record

SQL INSERT Query

SQL को INSERT INTO Statement को सहायताले डाटावेस टेवलको रोमा नयाँ डाटा थप्न सकिन्छ ।

Syntax: INSERT INTO statement का दुईवटा basic syntax हरु निम्न छन् ।

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)]VALUES (value1, value2, value3,...valueN);

जहाँ, column1, column2,...columnN हरु कोलमका नाम हुन् जसमा हामीले नयाँ डाटा थप्न खोज्दैछौँ । यदि सबै कोलममा डाटाहरु थप्दैछौँ भने कोलमको नाम दिनुपर्ने जरुरी हुँदैन । तर सबै डाटाको क्रम टेवलमा रहेको प्रत्येका कोलमको क्रमसँग मिलेको हुनु आवश्यक पर्दछ

SQL को INSERT INTO को syntax निम्न छ ।

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

Example: दिइएको कमाण्डले ६ वटा रेकर्डहरु CUSTOMERS table मा थप्न प्रयोग गरिएको छ :

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)VALUES (1, 'Ramesh', 32, 'Ghorahi', 2000.00);

OR

हामिले नयाँ रेकर्डहरु ***CUSTOMERS table*** थप्नको लागि निम्न कमाण्ड पनि प्रयोग गर्न सक्छौँ ।

INSERT INTO CUSTOMERSVALUES (7, 'Muffy', 24, 'Indore', 10000.00);

SQL UPDATE Query

The SQL UPDATE Query को सहायताले टेवलमा पहिले भएको रेकर्डहरु परिवर्तन गर्न सकिन्छ । यसको लागि UPDATE query सँग WHERE clause को प्रयोग गरी आवश्यक रो मात्र update गर्न सकिन्छ अन्यथा यसले सबै रेकर्डलाई update गर्दछ ।

Syntax: UPDATE query सँग WHERE clause को प्रयोग गर्ने basic syntax निम्न छ ।

UPDATE table_nameSET column1 = value1, column2 = value2...., columnN = valueNWHERE [condition];

हामीले आवश्यकता अनुसार धेरै conditions हरु AND or OR operators को सहायताले जोड्न सक्छौं ।

Example: मानौं EMPLOYEE table मा निम्न रेकर्डहरु छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

तल दिइएको कमाण्डले दिइएको टेवलको ADDRESS कोलम update गर्दछ जसको ID ६ रहेको छ ।

SQL> UPDATE CUSTOMERS SET ADDRESS = 'Pune' WHERE ID = 6;

अब, CUSTOMERS table मा निम्न रेकर्डहरु हुनेछन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

SQL DELETE Query

SQL DELETE Query को सहायताले टेवलमा रहेको रेकर्डहरु हटाउन सकिन्छ । हामीले DELETE query सँग WHERE clause को प्रयोग गरी आवश्यक रो मात्र delete गर्न सकिन्छ अन्यथा यसले सबै रेकर्डलाई delete गर्दछ ।

Syntax: DELETE query सँग WHERE clause प्रयोग गर्ने basic syntax निम्न छ ।

DELETE FROM table_nameWHERE [condition];

हामीले आवश्यकता अनुसार धेरै conditions हरु AND or OR operators को सहायताले जोड्न सक्छौं ।

Example: मानौं EMPLOYEE table मा निम्न रेकर्डहरु छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

तल दिइएको कमाण्डले CUSTOMERS table बाट एउटा customer को रेकर्ड हटाउँछ जसको ID ६ रहेको छ ।
SQL> DELETE FROM CUSTOMERSWHERE ID = 6;
अब, CUSTOMERS table तल दिइएको जस्तो देखिनेछ ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

यदि CUSTOMERS table का सबै रेकर्डहरु हटाउनु परेमा WHERE clause प्रयोग गर्नु पर्दैन । जसको लागि दिइएको कमाण्ड प्रयोग गर्नुपर्ने हुन्छ ।
SQL> DELETE FROM CUSTOMERS;
अब, CUSTOMERS table मा कुनै रेकर्डहरु पनि हुने छैनन् ।

SQL TRUNCATE TABLE

SQL TRUNCATE TABLE command को सहायताले कुनैपनि टेवलमा रहेका सबै डाटाहरु हटाउन सकिन्छ तर यसले टेबलको पूरै structure लाई भने हटाउदैन । यस कार्यका लागि DROP TABLE command पनि प्रयोग गर्न सकिन्छ तर DROP TABLE command ले टेवलको पूरै structure नै हटाउँछ जसले गर्दा यदि सो टेवल पुनः प्रयोग गर्नुपरेमा re-create गर्नुपर्ने हुन्छ ।

Syntax: TRUNCATE TABLE command को basic syntax निम्न छ ।
TRUNCATE TABLE table_name;

Example: मानौँ CUSTOMERS table मा निम्न रेकर्डहरु छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

माथिको टेवललाई truncate गर्नको लागि निम्न कमाण्ड प्रयोग गरिन्छ :
SQL > TRUNCATE TABLE CUSTOMERS;
अव, CUSTOMERS table truncate हुन जान्छ र SELECT statement प्रयोग गरी हेर्दा निम्न आउटपुट देखिन्छ ।

SQL> SELECT * FROM CUSTOMERS;
Empty set (0.00 sec)

Unit-5: Obtain reference Data

Reference data

SQL SELECT Query

SQL SELECT Statement को सहायताले डाटावेस टेवलबाट डाटा तान्न सकिन्छ जुन result table को रुपमा देखिने गर्दछ । यी result table हरुलाई result-sets भन्ने गरिन्छ ।

.

Syntax: SELECT statement को basic syntax निम्न छ ।

SELECT column1, column2, columnN FROM table_name;

जहाँ, column1, column2... हरु भनेको टेवलका फिल्डहरु हुन् जसबाट डाटा तान्न खोजिएको हो । यदि टेवलमा रहेका सबै फिल्डहरुबाट डाटा तान्नु परेमा निम्न syntax प्रयोग गर्न सकिन्छ ।

SELECT * FROM table_name;

Example: मानौं CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

निम्न दिइएको उदाहरणमा हामीले CUSTOMERS table बाट customers हरुको ID, Name and Salary fields तान्न खोजिएको छ ।

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

यस कमाण्डले निम्न रिजल्ट प्रदान गर्नेछ ।

ID	NAME	SALARY
1	Ramesh	2000.00
2	Khilan	1500.00
3	kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

Conditional reference with WHERE:

SQL WHERE Clause

The SQL WHERE clause को प्रयोग condition राखेर एक वा एकभन्दा बढी टेवलहरुबाट डाटा तान्नको लागि पयोग गरिन्छ । यदि मिलेछ भने मात्र यसले निश्चित हरु टेवलबाट दिने गर्दछ । यसमा रेकर्डहरु filter गर्नको लागि WHERE clause को प्रयोग गरिन्छ जसले आवश्यक रेकर्डहरुमा मात्र तान्ने काम गर्दछ । WHERE clause, SELECT statement मा मात्र नभई UPDATE, DELETE statement आदिमा पनि प्रयोग गर्न सकिन्छ ।

Syntax: SELECT statement सँग WHERE clause को basic syntax निम्न छ ।
SELECT column1, column2, column FROM table_name WHERE [condition]

हामीले आवश्यकता अनुसार You can specify a condition using comparison वा logical operators जस्तै >, <, =, LIKE, NOT आदि प्रयोग गरेर पनि condition बनाउन सकिन्छ । तल दिइएको उदाहरणले यसलाई अझ स्पष्ट पार्नेछ ।

Example: मानौँ CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

तल दिइएको उदाहरणमा हामीले CUSTOMERS table बाट ID, Name and Salary field हरु तान्ने छौं जसको salary 2000 वा 2000 भन्दा बढी रहेको छ ।

SQL> SELECT ID, NAME, SALARYFROM CUSTOMERSWHERE SALARY > 2000;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ ।

ID	NAME	SALARY
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

SQL Joins

SQL Joins clause को प्रयोग डाटावेसमा रहेका दुई वा दुईभन्दा बढी टेवलहरुको रेकर्डहरु मिलाउनको लागि प्रयोग गरिन्छ । JOIN एउटा माध्यम हो जसको सहायताले दुईवटा टेवलका फिल्डहरुमा रहेका समान values को आधरमा रेकर्डहरु मिलाउने कार्य गरिन्छ । is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

मानौ तल दिइएको दुईवटा टेवलहरु मध्ये (a) CUSTOMERS table मा निम्न रेकर्डहरु छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

(b) अर्को ORDERS table मा निम्न रेकर्डहरु छन् ।

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

अब, यी दुई टेवलहरु जोड्नको लागि SELECT statement को प्रयोग निम्न छ ।

SQL> SELECT ID, NAME, AGE, AMOUNT FROM CUSTOMERS, ORDERS WHERE CUSTOMERS.ID
= ORDERS.CUSTOMER_ID;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ ।

ID	NAME	AGE	AMOUNT
3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

यहाँ ध्यान दिनुपर्ने कुरा के छ भने यसमा जोड्नको लागि WHERE clause को प्रयोग गरिएको छ । टेवलहरु जोड्नको लागि विभिन्न operators हरु जस्तै =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT पनि प्रयोग गर्न सकिन्छ । यद्यपी सबैभन्दा बढी प्रयोग हुने operator, '=' हो ।

SQL Group By

The SQL GROUP BY clause को प्रयोग SELECT statement मा सहयोगी statement को रुपमा प्रयोग गरिन्छ जसले समान डाटाहरुको समूह बनाएर क्रमबद्धरुपमा देखाउने काम गर्दछ । GROUP BY जहिलेपनि WHERE clause पछि SELECT statement सँग प्रयोग हुन्छ र अन्त्यमा ORDER BY clause प्रयोग गरिन्छ ।

Syntax: GROUP BY clause को basic syntax निम्न छ ।
The GROUP BY clause जहिले पनि WHERE clause मा रहेको conditions पछि हुनुपर्दछ र त्यसपछि क्रम मिलाउनको लागि ORDER BY clause को प्रयोग गर्नुपर्दछ ।

SELECT column1, column2FROM table_name WHERE [conditions] GROUP BY column1, column2
ORDER BY column1, column2

Example: मानौं CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

यदि हामीलाई प्रत्येक customer को total amount of salary कति छ थाहा पाउनको लागि GROUP BY query निम्न हुनुपर्दछ ।
SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS GROUP BY NAME;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ ।

NAME	SUM(SALARY)
Chaitali	6500.00
Hardik	8500.00
kaushik	2000.00
Khilan	1500.00
Komal	4500.00
Muffy	10000.00
Ramesh	2000.00

The SQL ORDER BY

The SQL ORDER BY clause को प्रयोग डाटाहरुलाई एक वा एकभन्दा बढी कोलमहरुको आधारमा (सानोदेखि ठूलो वा ठूलो देखि सानो) क्रमबद्ध मिलाउनको लागि प्रयोग गरिन्छ । कुनै डाटावेसले सामान्यतया सानो देखि ठूलो क्रममा मिलाएर देखाउने गर्दछन् ।

***Syntax:* ORDER BY clause को basic syntax निम्न छ।**

SELECT column-list FROM table_name [WHERE condition] [ORDER BY column1, column2, .. columnN] [ASC | DESC];

You can use more than one column in the ORDER BY clause मा आवश्यकताअनुसार एक भन्दा बढी कोलमहरुको प्रयोग गर्न सकिन्छ। जुनसुकै कोलमको क्रम मिलाउँदा पनि उक्त कोलम column-list मा छ भन्ने कुरा यकिन गर्नुपर्ने हुन्छ।

Example: मानौँ CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन्।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

तल दिइएको उदाहरणमा प्राप्त रिजल्ट NAME र SALARY को आधारमा सानो देखि ठूलो क्रममा मिलाएर देखाउने कार्य गर्दछ।

SQL> SELECT * FROM CUSTOMERSORDER BY NAME, SALARY;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ।

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
3	kaushik	23	Kota	2000.00
2	Khilan	25	Delhi	1500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
1	Ramesh	32	Ahmedabad	2000.00

SQL Sub Queries

Sub query वा Inner query वा Nested query भनेको त्यस्तो query हो जुन अर्को SQL query भित्र प्रयोग गरिन्छ र जसलाई WHERE clause सँग जोडिएर सँगसँगै प्रयोग गरिन्छ।

Sub query एउटा main query को condition को रुपमा रहेको हुन्छ जसले डाटालाई थप सीमित गर्दै डाटाहरु निकाल्ने गर्दछ।

sub queries हरु सँग SELECT, INSERT, UPDATE, and DELETE statements र यी statements सँग operators हरु जस्तै like =, <, >, >=, <=, IN, BETWEEN पनि राख्न सकिन्छ।

Sub queries with the SELECT Statement:

Sub queries हरु धेरैजसो SELECT statement सँग प्रयोग गरिने गरिन्छ । Basic syntax निम्न रहेको छ ।

SELECT column_name [, column_name] FROM table1 [, table2] WHERE column_name OPERATOR (SELECT column_name [, column_name] FROM table1 [, table2] [WHERE])

Example: मानौँ CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

अब, निम्न sub query लाई SELECT statement सँग जाँच गर्दा ।

SQL> SELECT *FROM CUSTOMERSWHERE ID IN (SELECT IDFROM CUSTOMERS WHERE SALARY > 4500) ;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ ।

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

SQL Unions Clause

The SQL UNION clause/operator को प्रयोग दुई वा दुईभन्दा बढी SELECT statements बाट आएको नतिजालाई जोडेर, नदोहोराइ रेकर्डलाई देखाउनको लागि प्रयोग गरिन्छ ।

UNION प्रयोग गर्नको लागि प्रत्येक SELECT statement मा समान सँख्यामा कोलम छानिएको हुनुपर्दछ, समान सँख्यामा कोलम expression हुनुपर्दछ, समान data type र समान क्रम भएको हुनुपर्दछ तर तिनीहरुको लम्बाइ भने फरक फरक हुनसक्छ ।

Syntax: UNION को basic syntax निम्न छ ।

SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition] UNION SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition]

यहाँ दिइएको condition हाम्रो आवश्यकता अनुसारको जुनसुकै expression हुनसक्छ ।

Example: मानौ तल दिइएको दुईवटा टेवलहरु मध्ये (a) CUSTOMERS table मा निम्न रेकर्डहरु छन् ।

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

(b) अर्को ORDERS table मा निम्न रेकर्डहरु छन् ।

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

अब, तल दिइएको SELECT statement प्रयोग गरी दिइएका दुइवटा टेवललाई जोडेर हेर्दा :

SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS LEFT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID UNION SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS RIGHT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

यस कमाण्डबाट निम्न नतिजा प्राप्त हुनेछ ।

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

Unit-6: Familiarize the Operators

Operators

SQL Operators

What is an Operator in SQL?

Operator भनेको एउटा reserved word वा एउटा character हो जुन मुख्यतया SQL statement को WHERE clause सँग प्रयोग गरिन्छ जसको सहायताले विभिन्न कार्यहरु गर्न सकिन्छ जस्तै : comparisons र arithmetic operations.

Operators को प्रयोग गरेर हामीले SQL statement मा condition दिन सक्छौँ वा एउटै statement मा धेरै conditions हरु जोडेर लेख्नको लागि पनि प्रयोग गर्न सक्छौँ ।

Types of operators: (operators का प्रकार)

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

SQL Arithmetic Operators:

SQL arithmetic operator को प्रयोग दुई वा दुई भन्दा भढि भेरिएबलका मानहरुलाई जोड, घटाउ, भाग गुणान, शेष आउने भाग जस्ता कार्य गरेर एउटा नयाँ नतिजा निकाल्नको लागि प्रयोग गरिन्छ । मानौ variable a को मान 10 र variable b को मान 20 रहेको छ भने,

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0

यहाँ SQL Arithmetic Operators को सामान्य रुपमा प्रयोग गर्ने उदाहरण दिइएको छ ।

```
SQL> select 10+ 20;
+---------+
| 10+ 20 |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)
```

```
SQL> select 10 * 20;
+----------+
| 10 * 20 |
+----------+
|      200 |
+----------+
1 row in set (0.00 sec)
```

SQL Comparison Operators: SQL comparison operator को प्रयोग दुई वा दुई भन्दा बढि भेरिएबलको मानहरुलाइ तुलना गरि सर्तको आधारमा एउटा नयाँ नतिजा निकाल्नको लागि प्रयोग गरिन्छ। मानौ variable a को मान 10 र variable b को मान 20 रहेको छ भने,

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

यहाँ SQL Comparison Operators को सामान्य रुपमा प्रयोग गर्ने उदाहरण दिइएको छ।

```
SQL> SELECT * FROM CUSTOMERS WHERE SALARY > 5000;
+----+----------+-----+---------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------+
|  4 | Chaitali |  25 | Mumbai  |  6500.00 |
|  5 | Hardik   |  27 | Bhopal  |  8500.00 |
|  7 | Muffy    |  24 | Indore  | 10000.00 |
+----+----------+-----+---------+----------+
```

```
SQL>  SELECT * FROM CUSTOMERS WHERE SALARY = 2000;
+----+---------+-----+----------+---------+
| ID | NAME    | AGE | ADDRESS  | SALARY  |
+----+---------+-----+----------+---------+
|  1 | Ramesh  |  32 | Ahmedabad | 2000.00 |
|  3 | kaushik |  23 | Kota     | 2000.00 |
+----+---------+-----+----------+---------+
```

```
SQL>  SELECT * FROM CUSTOMERS WHERE SALARY != 2000;
+----+----------+-----+---------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------+
|  2 | Khilan   |  25 | Delhi   |  1500.00 |
|  4 | Chaitali |  25 | Mumbai  |  6500.00 |
|  5 | Hardik   |  27 | Bhopal  |  8500.00 |
|  6 | Komal    |  22 | MP      |  4500.00 |
|  7 | Muffy    |  24 | Indore  | 10000.00 |
+----+----------+-----+---------+----------+
```

SQL Logical Operators:

यहाँ SQL मा उपलब्ध logical operators हरुको सूची दिइएको छ ।

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

यहाँ SQL Comparison Operators को सामान्य रुपमा प्रयोग गर्ने उदाहरण दिइएको छ ।

```
SQL> SELECT * FROM CUSTOMERS WHERE AGE >= 25 AND SALARY >= 6500;
+----+----------+-----+---------+---------+
| ID | NAME     | AGE | ADDRESS | SALARY  |
+----+----------+-----+---------+---------+
|  4 | Chaitali |  25 | Mumbai  | 6500.00 |
|  5 | Hardik   |  27 | Bhopal  | 8500.00 |
+----+----------+-----+---------+---------+
```

```
SQL> SELECT * FROM CUSTOMERS WHERE AGE >= 25 OR SALARY >= 6500;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Unit-7: Manage views

Manage Views

SQL - Using Views

View भनेको SQL statement हरु हुन् जुन डाटावेसमा सम्बन्धित नाम दिएर भण्डार गरेर राखिएको हुन्छ । वास्तवमा View भनेको टेवलको संरचना हो जुन पूर्वनिर्धारित SQL query को रुपमा रहेको हुन्छ ।

View मा टेवलमा रहेका रो हरु हुन सक्छन् वा टेवलबाट छानिएका रो हरु हुन सक्छन् । View एक वा एकभन्दा बढी टेवलहरुबाट पनि बनाउन सकिन्छ, जुन view बनाउनको लागि लेखिएको SQL query मा भर पर्दछ ।

Views, भनेको एक किसिमको काल्पनिक टेवलहरु हुन्, जसबाट प्रयोगकर्ताले निम्न कार्य गर्न सक्छन् ।

- डाटाको संरचना तयार गर्न सक्छन् जुन प्रयोगकर्ता वा विभिन्न वर्गका प्रयोगकर्तालाई स्वाभाविक लागोस् ।
- डाटामा नियन्त्रण राख्न सकिन्छ, जसले गर्दा प्रयोगकर्ताले निश्चित किसिमको डाटा मात्र हेर्न वा परिवर्तन गर्न पाउँछन् ।
- विभिन्न टेवलहरुको आधारमा डाटालाई सँक्षेपमा प्रस्तुत गर्न सकिन्छ जसबाट reports बनाउन सकिन्छ ।

Creating Views:

Database views बनाउनको लागि CREATE VIEW statement प्रयोग गरिन्छ । Views एउटा टेवलको आधारमा, धेरै टेवलको आधारमा वा अर्को view को आधारमा पनि बनाउन सकिन्छ । बनाउनको लागि सर्वप्रथम प्रयोगकर्तासँग उपयुक्त कार्य कार्यान्वयन गर्नको लागि विशेष अधिकार हुनु जरुरी हुन्छ ।

The basic CREATE VIEW syntax is as follows:

CREATE VIEW view_name AS SELECT column1, column2.....FROM table_name WHERE [condition];

SQL SELECT query प्रयोग गरे जस्तै गरेर माथिको SELECT statement मा पनि आवश्यकता अनुसार एकभन्दा बढी टेवलहरु पनि प्रयोग गर्न सक्छौँ ।

Example: मानौँ CUSTOMERS table मा निम्न रेकर्डहरु रहेका छन् ।

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
```

1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

अब, दिइएको CUSTOMERS table को आधारमा view बनाउन निम्न कमाण्ड प्रयोग गर्नुपर्ने हुन्छ । यस कमाण्डले CUSTOMERS table बाट customer name र age प्रयोग गरेर view तयार गर्दछ ।

SQL > CREATE VIEW CUSTOMERS_VIEW ASSELECT name, age FROM CUSTOMERS;

अब, माथि दिइएको कमाण्डको CUSTOMERS_VIEW आउटपुट हेर्न अन्य टेवलको आउटपुट हेरे जस्तै गरेर गर्न सकिन्छ । तल यसको उदाहरण दिइएको छ ।

SQL > SELECT * FROM CUSTOMERS_VIEW;

Dropping Views:

निश्चय रुपमा, कहिलेकाही हामीले बनाएको view अनावश्यक भएमा हटाउनुपर्ने पनि हुनसक्छ । view हटाउने सजिलो syntax तल दिइएको छ ।

DROP VIEW view_name;

CUSTOMERS table मा रहेको CUSTOMERS_VIEW हटाउनको लागि निम्न कमाण्ड प्रयोग गर्न सकिन्छ । :

DROP VIEW CUSTOMERS_VIEW;

Prepare index

INDEX:

INDEX ले डाटावेसमा डाटा छिटो राख्न वा प्राप्त गर्न मद्दत गर्दछ । बनाउनको लागि टेवलमा रहेको एक वा एकभन्दा बढी कोलमको प्रयोग गर्न सकिन्छ । जब Index बनाइन्छ, तब डाटाहरुको क्रम मिलाउनको लागि प्रत्येक रो को लागि ROWID निर्धारित गरिन्छ ।

ठूलो डाटावेसको लागि उपयुक्त indexes हरु प्रभावकारी मानिन्छ तर Index बनाउँदा विशेष ध्यान दिनुपर्ने हुन्छ । Index को लागि field छान्दा SQL queries को प्रयोगको आधारमा छान्दा उपयुक्त हुन्छ ।

Example:

उदाहरणको लागि, तलको SQL statement ले एउटा नयाँ टेवल CUSTOMERS निर्माण गर्दछ र त्यसमा ५ वटा कोलम थप्दछ ।

CREATE TABLE CUSTOMERS(ID INTNOT NULL, NAME VARCHAR (20) NOT NULL, AGE INT NOT NULL,ADDRESS CHAR (25) ,SALARY DECIMAL (18, 2), PRIMARY KEY (ID));

अब, हामीले निम्न syntax प्रयोग गरेर आवश्यकता अनुसार एक वा एकभन्दा बढी कोलमको प्रयोग गरेर index निर्माण गर्न सक्छौँ ।

CREATE INDEX index_nameON table_name (column1, column2.....);

यदि AGE column मा INDEX राखेर उमेरको आधारमा छिटो डाटाहरु खोज्नुपरेमा निम्न SQL कमाण्ड प्रयोग गर्नुपर्ने हुन्छ ।

syntax:
CREATE INDEX idx_age ON CUSTOMERS (AGE);
DROP an INDEX Constraint:
यदि INDEX constraint drop गर्नुपरेमा निम्न SQL statement प्रयोग गर्न सकिन्छ :
ALTER TABLE CUSTOMERSDROP INDEX idx_age;

Procedure

What is a procedure in MySQL?

A procedure भनेको डाटावेसमा रहेको stored program हो जसमा हामीले parameters दिन सक्छौँ । यसले function ले जस्तो value फिर्ता गर्दैन ।

Create Procedure

अन्य language मा procedures बनाए जस्तै गरेर MySQL मा पनि हामीले आफूलाई आवश्यक procedures बनाउन सक्छौ ।

Syntax:

MySQL मा procedure बनाउनको लागि निम्न syntax प्रयोग गर्न सकिन्छ ।

```
    CREATE PROCEDURE procedure_name [ (parameter datatype [,
    parameter datatype]) ]
    BEGIN
   declaration_section
   executable_section
END;
```

procedure_name भनेको MySQL मा procedure लाई दिइने नाम हो । *.(parameter datatype [, parameter])* भनको एक वा एकभन्दा बढी parameters हरु हुन् जुन हामीले procedure मा पठाउने गर्दछौँ । procedure बनाउँदा ३ किसिमका parameters हरु हामीले दिन सक्छौ ।

1. **IN** - Procedure ले parameter लाई reference को रुपमा लिने गर्दछ । procedure ले parameter को value लाई overwritten हुन दिँदैन ।
2. **OUT** - Procedure ले parameter लाई reference को रुपमा लिने गर्दैन तर procedure ले parameter को value लाई overwritten गर्दछ ।
3. **IN OUT** - Procedure ले parameter लाई reference को रुपमा लिने गर्दछ र procedure ले parameter को value लाई overwritten गर्दछ ।

declaration_section भनेको procedure को यस्तो ठाउँ हो जहाँ हामीले local variable हरु declare गर्न सक्छौँ ।
executable_section भनेको procedure को यस्तो ठाउँ हो जहाँ हामीले the procedure को लागि code लेख्ने गर्दछौँ ।

Drop procedure

MySQL मा procedure बनाइसकेपछि, आवश्यकता अनुसार हामीले त्यसलाई डाटावेसबाट हटाउन पनि सक्छौँ ।

Syntax

MySQL मा procedure लाई drop गर्नको लागि निम्न syntax प्रयोग गर्न सकिन्छ ।
`DROP procedure [ IF EXISTS ] procedure_name;`
procedure_name भनेको त्यो procedure को नाम हो जसलाई हामीले drop गर्न चाहन्छौँ ।

Example

तलको उदाहरणमा MySQL मा procedure लाई drop गर्ने तरिका दिइएको छ ।

For example:

```
DROP procedure CalcIncome;
```

माथिको उदाहरणमा MySQL ले *CalcIncome* भन्ने procedure लाई drop गर्दछ ।

Unit-8: Perform Transaction

Transaction

Introduction: Transaction भनेको database processing को त्यस्तो logical unit हो जसले डाटावेसमा विभिन्न कार्यहरु जस्तै : insertion, deletion, modification, or retrieval operations गर्दछ । डाटावेसमा कार्य गर्नको लागि बनाइएका transaction हरु सँग गाँसिएका हुन सक्छन् वा तिनीहरु विशेष किसिमको high-level query language जस्तै SQL बाट बनाइएका हुन सक्छन् ।

Desirable Properties of Transactions (Transactions का गुणहरु/विशेषताहरु):

Transaction मा मुख्य रुपमा ४ वटा गुणहरु हुनुपर्दछ ।: atomic, consistent, isolated and durable. यी database transactions का गुणहरुलाई कहिलेकाहिँ छोटो रुपमा **ACID** भनेर पनि चिनिने गरिन्छ ।

(a) ***Atomicity:***

Transaction भनेको स्वचालित हुने processing को unit हो, जुन या त पूर्ण रुपमा सँचालित हुन्छ वा हुँदैन ।

(b) ***Consistency preservation:***

Transaction ले सुनिश्चितता ग्रहण गरेको हुन्छ । यसको सञ्चालनबाट डाटावेसलाई यउटा स्थितिबाट अर्को स्थितिमा परिवर्तन गर्न सहयोग पुग्दछ ।

(c) Isolation:

Transaction जहिले पनि अन्य भन्दा अलग हुनु पर्दछ । जसको अर्थ एउटा transaction ले अर्को transaction को कार्यमा बाधा पुऱ्याउनु हुँदैन ।

(d) ***Durability or permanency:***

Transaction मार्फत डाटावेसमा गरिएको परिवर्तन स्थायी हुनुपर्दछ । यस्ता परिवतर्नहरु कुनै को कारणले हराएर जाने हुनुहुँदैन ।

Types of Transaction: (Transaction का प्रकार)

Type	Description
Data manipulation language (DML)	यसमा धेरै संख्यामा DML statements रहेका हुन सक्छन् जसलाई MySQL server ले एउटा entity वा logical unit को कार्यको रुपमा व्यवहार गर्दछ ।
Data definition language(DDL)	यसमा एउटा मात्र DDL statement रहेको हुन्छ ।
Data control language(DCL)	यसमा एउटा मात्र DCL statement रहेको हुन्छ ।

Transaction Control:

Transactions लाई नियन्त्रण गर्नको लागि निम्न कमाण्डहरु प्रयोग गरिन्छ ।:

- **COMMIT:** परिवर्तन गरिएको कुरा सेभ गर्नको लागि
- **ROLLBACK:** परिवर्तन गरिएको कुरा हटाउनको लागि
- **SAVEPOINT:** ROLLBACK गर्ने groups of transactions भित्र points निर्माण गर्नको लागि
- **SET TRANSACTION:** transaction मा नाम राख्नको लागि

Transactional control command हरु जहिलेपनि DML commands हरु जस्तै INSERT, UPDATE and DELETE सँग मात्र प्रयोग गरिन्छन् । यि कमाण्डहरुलाई टेवल बनाउने बेलामा वा हटाउने बेलामा प्रयोग गरिँदैनन् किनभने यी कार्यहरु डाटाबेस भित्र स्वचालित रुपमा हुन्छन् ।

- ***BEGIN TRANSACTION*:** यसले transaction को connection को लागि starting point चिन्ह लगाउँछ ।

- ***COMMIT TRANSACTION*:** यदि कुनै समस्या नआएमा यसले transaction सफलतापूर्वक अन्त्य गर्दछ । transaction ले परिवर्तन गरिएको सबै डाटालाई स्थायीरुपमा डाटावेसमा राख्छ । transaction ले प्रयोग गरेका Resources खाली गरिन्छ ।
- ***ROLLBACK TRANSACTION*:** समस्या आएका transaction लाई हटाउने कार्य गर्दछ । Transaction ले परिवर्तन गरेका सबै डाटाहरु पुरानै स्थितिमा फिर्ता ल्याइन्छ । Transaction ले प्रयोग गरेका Resources खाली गरिन्छ ।
- ***The `SAVEPOINT`statement:*** SAVEPOINT Statement लाई transaction को savepoint को लागि `identifier` को नाम सेट गर्दछ । यदि अहिले प्रयोग भएको transaction को savepoint को नाम उही भएमा पुरानो savepoint हटाइन्छ र नयाँ savepoint सेट गरिन्छ

Unit-9: Familiarize with functions

SQL Aggregate Functions

Useful aggregate functions:

- AVG() - यसले average value दिने गर्दछ ।
- COUNT() - यसले number of rows दिने गर्दछ ।
- FIRST() - यसले first value दिने गर्दछ ।
- LAST() - यसले last value दिने गर्दछ ।
- MAX() - यसले largest value दिने गर्दछ ।
- MIN() - यसले smallest value दिने गर्दछ ।
- SUM() - यसले sum दिने गर्दछ ।

String functions:

TRIM function ले SQL मा रहेको string बाट prefix या suffix हटाउने कार्य गर्दछ । सबैभन्दा बढी यसको प्रयोग खाली ठाउँ हटाउनको लागि प्रयोग हुन्छ । यो function डाटाबेस अनुसार फरक फरक तरिकाले प्रयोग गरिने गरिन्छ ।

MySQL: TRIM( ), RTRIM( ), LTRIM( )

TRIM([[LOCATION] [remstr] FROM] str)

[LOCATION] मा LEADING, TRAILING, वा BOTH हुन सक्छन् । यो function ले remstr] pattern हरु string को सुरु वा अन्त्यबाट हटाउँछ । यदि [remstr] दिइएन भने यसले खाली ठाउँ हटाउँछ ।

TRIM(): यसले अगाडि र पछाडिका spaces हटाउँछ ।

Syntax: TRIM([*characters* FROM]*string*)

Example: SELECT TRIM('#! ' FROM ' #SQL Tutorial! ') AS TrimmedString;

LTRIM(): LTRIM को प्रयोग अगाडिका spaces हटाउँनका लागि प्रयोग गरिन्छ ।

Syntax: LTRIM(*string*)

Example: SELECT LTRIM(' SQL Tutorial') AS LeftTrimmedString;

RTRIM(): पछाडिका spaces हटाउँनको लागि RTIM() को प्रयोग गरिन्छ ।

Syntax: RTRIM(*string*)

Example: SELECT RTRIM('SQL Tutorial ') AS RightTrimmedString;

Practical Example:

1. To create a table bill.

```
(use joinexample database)
create table bill
(
        sn int not null,
        particular varchar(30),
```

```
        qty varchar(30),
        rate int,
        remarks varchar(30),
)
```

Press **F5** for run.

2. To see the table in full view

```
        select * from bill
```

3. To inset records in table bill.

```
insert into bill(sn,particular,qty,rate,remarks)
values(1,'book',12,440,'good')
```

4.To create table **address** having following columns:

```
create table address
(
        id int not null,
        name varchar(25),
        Address varchar(30),
        Phone int,
        Email varchar(40),
        Remarks varchar(25),
);
```

5. To insert the columns **lname** and **nickname** into a table address.

```
        alter table address
        add lname varchar(20), nickname varchar(10);
```

6. To delete column **nickname** from table address.

```
        alter table address
        drop column nickname;
```

7. To delete row 2 (id=2) from table address.

```
        delete from address
        where id=2;
```

8. To display the **names** in ascending order from table address.

```
        select name
        from address
        order by name asc;
```

9. To display **address** from table address.

```
        select address
        from address;
```

10. To display records from table address whose address are in Pyuthan

```
        select *from address
        where address ='Pyuthan'
```

11. To count the number of the columns of table address.

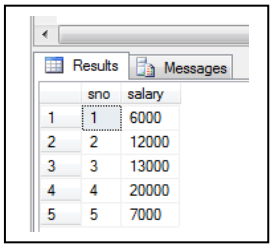
```
        select count(*)
        from address;
```

12. To display maximum value of the column **salary** from table **job.**

```
select max(salary)
from job
```

Eg.

```
create table job
(
sno int not null,
salary int,
)
```



	sno	salary
1	1	6000
2	2	12000
3	3	13000
4	4	20000
5	5	7000

13. To display minium value of the column **salary** from table **job.**

```
select min(salary)
from job
```

14. To display sum of the column **salary** from table **job.**

```
select sum(salary)as Total
from job
```

15. To display average value of the column **salary** from table **job.**

```
select avg(salary)
from job
```

16. Comparison Predicate :

To display the salary more than 5000 from table job & column salary.

```
select salary
from job
where salary >8000
```

17. Logical Comparison Predicates :

To display the salary more than 4000 and less than 20000 from table job & column salary.

```
select salary
from job
where salary >4000 and salary <15000
```

18. Between:

To display the salary between 4000 and 20000 & only salary column from table job.

```
select salary
from job
where salary between 4000 and 15000
```

19. To display the salary between 4000 and 20000 & all table salary column from table job.

```
select * from job
where salary between 4000 and 15000
```

20. **in :**

a. To display the data (20000) of column salary using **in** from table job.

```
select *
from job
where salary in(20000)
```

b. To display the name **Gita and Shyam** from table address & column name, all column.

```
select *
from address
where name in('Gita','Shyam')
```

c. To display the name **Gita and Shyam** from table address & column name, only name column.

```
select name
from address
where name in('Gita','Shyam')
```

21. To display name 'Gita' from table address by using subquery.

```
select *
from address
where name in(select name from address where name ='Gita')
```

22. To display salary =20000 from table job using subquery.

```
select *
from job
where salary in(select salary from job where salary =20000)
```

23. a. Using Primary Key:

```
Create table Eppy
(
Cno int constraint cno primary key,
Age int,
);
```

OR

```
create table Eppy1
(
cno int,
age int,
constraint cno_pk primary key (cno)
);
```

b. Composite Primary Key:

```
create table res
(
id int,
age int,
constraint id_pk primary key (id,age )
);
```

24. To display constraint name and detail information of the table.

```
exec sp_help Eppy
```

25. To remove the primary key from the table Eppy1.

```
alter table Eppy1
drop constraint cno_pk
```

selet all the student whose faculty is computer and year is 3rd

```
select *from student1
where faculty ='computer' and year=3
```

36. To add the primary key in the table Eppy1

```
alter table Eppy1
```

```
add constraint cno_pk primary key(cno)
```

37. **Unique Key:** It is similar to Primary Key, but it has following differences.

```
create table ramesh
(
sno int,
address varchar(10),
constraint sno_uk unique (sno),
);
```

38. Create Index :

```
create index sno_ind on job (sno)
```

39. To show the index type on the table.

```
exec sp_helpindex job
```

40. To remove the index from the table

```
drop index job.sno_ind (where job is a table name)
```

41. **Identity Property:** is the value that is used for very fast loaded into the table. To add incremental value i.e. add to the identity value of the previous that was loaded. If we do not specify then default is (1,1)

```
create table roshan
(
sno int identity(2,4),
name varchar(20)
);
insert into roshan (name) values('rajan')
insert into roshan (name) values('raju')
select * from roshan
```

sno	**name**
2	rajan
6	raju

42. Default :

```
create table himal
(
sno int default 10,
name varchar(20),
);
insert into himal (name) values('eppy')
insert into himal (name) values('raju')

select *from himal
```

sno	**name**
10	eppy
10	raju

43. By default the **select** clause includes duplicate values. If we want to force the elimination of duplicates the **distinct** keyword is used as follows:

```
select distinct branch-name
from Loan
```

23. To display the no of sno greater than 2 using **having and count** statement from table job.

```
select sno
from job
group by sno
having count(*) >=2
```

24. **Exists:**

```
select *from r as r
where exists
(select *from job where job.sno=r.sno)
select *from r
where exists
(select *from job where job.sno=r.sno)
```

25. Join(inner join) : on the basis of sno of table R and Job.

```
select r.sno,r.name,job.salary
from r,job
where r.sno=job.sno
```

26. Join(Natural join): It joins two tables. It shows all columns of one table and apart of another table which is specified by the user.

```
select  job.*,r.name
from r,job
where r.sno=job.sno
```

27. Joins Non-equality : It joins both the tables which donot satisfied the condition. It shows the output in cross product principle.

```
select  job.*,r.name
from r,job
where r.sno!=job.sno
select tt.cno,r.name
from r,tt
where cno!=sno
```

28. **Outer join** :

```
select tt.cno,r.name
from r full outer join tt
ON
cno=sno
```

29. Left join : It joins both table but it shows the data of left table.

```
select tt.cno,r.name
from r left outer join tt
ON
cno=sno
```

30 Right join .It joins the both tables but shows the data of right table.

```
select tt.cno,r.name
from r right outer join tt
ON
cno=sno
```

31. Declare statemet :

```
declare @ram varchar(10), @giri varchar(10)
set @ram ='master'
set @giri='address'
exec ('use ' +@ram+' select * from '+@giri)
```

32. Equijoin : The equijoin joins two tables with a common column in which each is equally the primary key.

```
select job.sno,r.name,job.salary
from job,r
where job.sno=r.sno
```

insert into tmp values(1,’ram’) run it Press F5

Message comes: INSERT statement conflicted with COLUMN CHECK constraint 'CK__tmp__st_id__23A93AC7'. The conflict occurred in database 'master', table 'tmp', column 'st_id'.

The statement has been terminated.

insert into tmp values(6,’ram’) run it Press F5

Display Records

Select *from tmp

b) Check constrains

```
create table tmp
(
st_id int constraint st_id_chk check(st_id>5),
name varchar(20))
```

Run Press F5 to run

insert into tmp values(1,'ram')

Error message comes:

INSERT statement conflicted with COLUMN CHECK constraint 'st_id_chk'. The conflict occurred in database 'master', table 'tmp', column 'st_id'.

The statement has been terminated.

Insert into tmp values(6,’ram’)

Insert into tmp(name) values(‘hari’)

Select *from tmp Press F5

Note: To display name of the constrains which are used in the table.

Example :

exec sp_help tmp Press F5.

It display the message:

To display information about the columns i.e .
exec sp_columns tmp Press F5.

exec sp_helpconstraint tmp
47.Disabling Constraint Checking On Existing Data: It disable the constraint condition given on the table, after that we can insert any values.

Eg. alter table tmp nocheck constraint st_id_chk Press F5.
insert into tmp values(2,'shyam') Press F5.

48. Enable Constraints: To set constraint on the table, after disabling the constraint.
alter table tmp check constraint st_id_chk

Note:
Insert into tmp values(1,'ram')
Error message comes:

49. To drop constraint on the table.
Eg.
alter table tmp drop constraint st_id_chk Press F5.

50. Add constraint in existing table.
Eg.
alter table tmp with check
add constraint st_id_chk check(st_id>1)

Note:
Set the value greater than exits value on the table else it will not support while.

Familiarize with Database Systems

Data Models:
Introduction: data model भनेको conceptual design process को एउटा भाग हो । अन्य सामान्यतया functional model हुने गर्दछन् । data model ले के कस्ता डाटाहरु डाटावेसमा भण्डार हुने भन्ने विषयमा ध्यान केन्द्रित गर्दछ भने functional model ले कस्ता किसिमका डाटाहरु process गर्ने भन्ने कुरामा ध्यान केन्द्रित गर्दछ । यसलाई relational database को परिप्रेक्षमा भन्नुपर्दा यसको प्रयोग relational tables हरु design गर्नको लागि प्रयोग गरिन्छ । functional model को प्रयोग queries हरु design गर्नको लागि प्रयोग गरिन्छ जसबाट टेवलमा रहेका डाटाहरुलाई access गर्ने र तिनीहरुमा विभिन्न कार्य गर्नको लागि प्रयोग गरिन्छ ।

Definition: A data model भनेको conceptual tools को समूह हो जसले डाटालाई वर्णन गर्ने, डाटाको रिलेसनसिप, data semantics, and consistency constraints आदि को बारेमा जानकारी दिन्छ ।.

Types of data models:

***(a) Entity–relationship model*:** The entity–relationship (E-R) model एउटा high-level data model हो । यो वास्तविक अनुभवमा आधारित रहेको हुन्छ जसमा basic objects हरुको समूह रहेको हुन्छ जसलाई entities भन्ने गरिन्छ र तिनीहरुको विचमा सम्बन्ध स्थापित गरिएको हुन्छ ।

***(b) Relational model*:** The relational model एउटा lower-level data model हो । यो टेवलहरुको समूह हो जसले डाटा तथा तिनीहरुबिचको सम्बन्धलाई प्रतिनिधित्व गर्दछ ।

Function: Main functions of data model are: (data model का मुख्य कार्यहरु निम्न छन् ।)

- Data model को मुख्य कार्य भनको वास्तविक संसारको वातावरणमा रहेका जटिलतालाई बुझ्न मद्दत गर्नु हो ।
- Within the database environment भित्र, a data model ले डाटाको संरचना, तिनीहरुको विशेषता, सम्बन्ध, सिमा तथा रुपान्तरण लागि प्रतिनिधित्व गर्दछ । राम्रो database design ले जहिलेपनि उपयुक्त data model को जगमा बनेको हुन्छ ।
- data model ले डाटाको blueprint को रुपमा कार्य गर्दछ जुन functional system मा आवश्यक पर्दछ ।

Importance:Importance of data model can be listed in the following ways:

- Data models ले designer, the application programmer र end user बिच कुराकानी गर्न सहयोग गर्दछ ।
- एउटा राम्रो किसिमले बनाइएको data model ले कुनैपनि सस्थाको database design लाई बुझ्न अझ मद्दत गर्दछ ।

Levels of data abstraction:

Data abstraction भनेको आवश्यक विशेषताहरु कार्यान्वयन नगरेर मात्र प्रतिनिधित्व गर्ने प्रकृया हो । धेरैजसो database-systems का प्रयोगकर्ताहरु त्यति अनुभवि नहुने भएकाले developers हरुले डाटावेसको जटिलतालाई विभिन्न तहको abstraction अनुसार लुकाएर प्रयोगकर्तालाई system सँग सहज अन्तरक्रिया गर्न मद्दत गर्दछन् ।

1) Physical level.

सबैभन्दा lowest level को abstraction ले data वास्तविक रुपमा कसरी भण्डार गर्ने वर्णन गर्दछ । physical level ले जटिल low-level को डाटा संरचनालाई विस्तृत रुपमा वर्णन गर्दछ ।

2) Logical level.

दोस्रो higher level को abstraction ले के कस्ता डाटाहरु भण्डार भएका छन् र तिनीहरुबिचको कस्तो सम्बन्ध रहेको छ भन्ने कुरा वर्णन गर्दछ । त्यसकारण logical level ले सम्पूर्ण डाटावेसलाई विभिन्न साना सना सम्बन्धित संरचनाको रुपमा वर्णन गर्ने गर्दछ ।

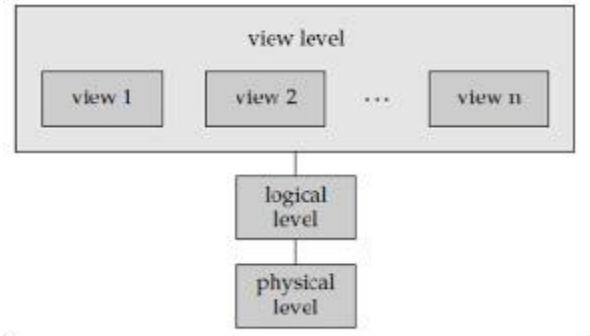


Figure 3 : Three Levels of Data Abstractions

3) View level.

सबभन्दा highest level को abstraction ले सम्पूर्ण डाटावेसको केही भाग मात्र वर्णन गर्ने गर्दछ । एउटा ठूलो डाटावेसमा विभिन्न थरीका सूचनाहरु भण्डार गरिएको हुन्छ । धेरैजसो database system का users हरुलाई सबै सूचनाहरु नचाहिने हुन्छ, त्यसकारण

उनीहरुलाई डाटावेसको केही भाग मात्र access गर्नुपर्ने हुन्छ । View level को abstraction ले users लाई system सँग interaction गर्न सहज गराउँदछ

Data independence: Data independence भनेको database management को एउटा रुप हो जसले डाटालाई सबै प्रोग्रमहरुबाट अलग राख्न मद्दत गर्दछ । यसले DBMS or database management system मा नयाँ सोचको विजारोपण गरेको छ, जसमा independence ले गर्दा कुनैपनि प्रोग्रामले डाटावेसको डाटालाई redefined वा reorganized गर्न नसकोस् खाली प्रयोग मात्र गर्न सकोस् भन्ने हो । यसबाट डाटा जहिले.पनि सुलभ हुन जान्छ र यो स्थिर पनि हुन्छ र कुनैपनि एप्लिकेसन प्रयोग्रामले यसलाई corrupt गर्न सक्दैन ।

Data independence दुई प्रकारका हुन्छन् ।

• Logical data independence
• Physical data independence

Concurrency control: Concurrency control एउटा database management systems (DBMS) को सोच हो जसले multi-user system मा लगातार रुपमा डाटालाई access वा alter गर्दा आइपर्ने conflicts हरुलाई सम्बोधन गर्दछ । Concurrency control जब DBMS मा लागु गरिन्छ तब यसले लगातार रुपमा हुने transactions हरुलाई समन्वय गर्दछ र साथसाथै data integrity पनि सम्हालेर राख्दछ । Concurrency भनेको डाटावेसमा हुने multi-user हरुको access लाई नियन्त्रण गर्ने बारेमा हो ।